# Матрица компетентности программиста (Programmer Competency Matrix)

С помощью данной таблицы вы можете оценить свой уровень знаний в каждой области. Уровни идут от 0 («начинающий») до 3 («эксперт»). Каждый следующий уровень включает в себя предыдущий , т.е. разработчик, находящийся на уровне 3, должен удовлетворять критериям двух предыдущих уровней.

Перевод взят с <https://dev.by/pages/programming_matrix>

Оригинал (на английском): <http://www.starling-software.com/employment/programmer-competency-matrix.html>

|  | Уровень 0 | Уровень 1 | Уровень 2 | Уровень 3 | Комментарии |
| --- | --- | --- | --- | --- | --- |
| Теория | | | | | |
|  | Уровень 0 | Уровень 1 | Уровень 2 | Уровень 3 | Комментарии |
| Структуры данных | Не понимает разницы между массивом и связным списком. | Может объяснить и использовать на практике массивы, связные списки, словари и т.д. | Понимает плюсы и минусы использования тех или иных базовых структур данных (размер памяти, время выполнения операций с данными, в чем разница между массивами и связными списками в этом плане).Может объяснить как реализовать хэш-таблицы и как обработать коллизии. Приоритетные очереди и способы их реализации, и т.д. | Знание сложных структур данных, таких как B-дерево, Биномиальная куча и Фибоначчиевская куча, АВЛ-дерево, Красно-чёрное дерево, Косое дерево, Список с пропусками, TRIE-структуры и т.д. |  |
| Алгоритмы | Не может найти среднее значение массива чисел. (Тяжело поверить, но встречаются и такие кандидаты на собеседовании.) | Базовые методы сортировки и поиска. Обход и поиск в структурах данных. | Деревья, Графы, "простой путь" и "разделяй-и-властвуй" алгоритмы, понимает значимость уровней приведенной здесь матрицы. | Может распознать и написать решение динамическим программированием, хорошо знает алгоритмы на графах, хорошо знает численные методы, может идентифицировать проблемы класса NP. | Работать с теми, кто имеет хороший рейтинг в TopCoder - это невероятная удача! |
| Системное программирование | Не знает что такое компилятор, линковщик или интерпретатор. | Базовое понимание компиляторов, линковщиков и интерпретаторов.Понимает, что такое ассемблерный код и как работают программы на уровне железа. Небольшое понимание виртуальной памяти и пэйджинга. | Понимает чем отличается kernel mode от user mode, что такое мульти-трединг, способы синхронизации и как реализованы примитивы синхронизации, может читать ассемблерный код. Понимает, как работают сети, сетевые протоколы и может реализовать передачу данных через сокеты. | Понимает как работает весь "программный стэк": железо (CPU + Память + Кэш + Прерывания + микрокоды), двоичный код, ассемблер, статическая и динамическая линковка, копиляция, интерпретация, JIT-компиляция, сборка мусора, куча, стэк, адресация памяти….. |  |
| Навыки | | | | | |
|  | Уровень 0 | Уровень 1 | Уровень 2 | Уровень 3 | Комментарии |
| Контроль версий исходников | Бэкап исходников в папку с датой бэкапа. | VSS и основы CVS/SVN в качестве пользователя | Специалист по возможностям CVS and SVN. Знает как разветвить и слить, настроить репозитарий и т.д. | Знает распределенные системы VCS. Пробовал Bzr/Mercurial/Darcs/Git |  |
| Автоматизация build'ов | Знает как запустить Build из среды программирования. | Умеет билдить из командной строки. | Может настроить скрипт для сборки основной системы. | Может настроить скрипт для сборки системы и документации, инсталляторов, генераторов "release notes" и добавит скрипт в систему контроля версий исходников. |  |
| Автоматизированное тестирование | Думает, что тестирование - это работа тестеров. | Написал автоматизированные юнит-тесты и может создавать свои хорошие юнит-тесты для кода, который пишет в настоящее время. | Пишет код в стиле Test-driven Development (TDD). | Понимает и может настроить автоматические тесты на функционал, пользовательский интерфейс и загрузку/производительность. |  |
| Программирование | | | | | |
|  | Уровень 0 | Уровень 1 | Уровень 2 | Уровень 3 | Комментарии |
| Декомпозиция задачи | Просто последовательные строчки кода, а copy/paste -для повторного использования кода. | Может разбивать решение задачи на несколько функций. | Способен создавать многократно используемые функции/объекты, которые решают общие задачи. | Использует соответствующие структуры данных и алгоритмы. Создает общий/объектно-ориентированный код, который инкапсулирует те условия задачи, которые могут быт изменены. |  |
| Декомпозиция системы | Не способен думать о системе, сложнее одного класса или файла. | Может произвести декомпозицию задачи и спроектировать систему в пределах одной платформы или технологии. | Может спроектировать систему, которая охватывает несколько технологий/платформ. | Может визуализировать и проектировать сложные системы с несколькими линейками продуктов и интеграцией с внешними системами. Также должен уметь проектировать системы поддержки работы: мониторинг, генерация отчетов, аварийные переходы на использование запасных ресурсов. |  |
| Общение | Не может выразить свои мысли/идеи. Плохо с правописанием и грамматикой. | Его понимают. Хорошие правописание и грамматика. | Может эффективно общаться. | Может понимать и объяснять мысли/дизайн/идеи/специфику в точно выраженной форме, в общении соответствует ситуации. | Важность этого критерия для программиста часто недооценивают. С увеличением аутсорсинга разработки ПО в те страны, где английский не является родным языком, этот вопрос стал более актуальным. Я знаю несколько проектов, которые провалились потому, что программисты не могли понять смысл обсуждения. |
| Организация кода в файле | Нет четкой организации в файле. | Методы сгрупированны логически и по вызовам. | Код разделен на регионы, имеет хорошие комментарии, в т.ч. со ссылками на другие файлы исходников. | Файл имеет разделы "license header", "summary", хорошие комментарии, непротиворечивую расстановку пробелов и табуляции. Файл должен выглядеть красиво. |  |
| Организация кода между файлами | Не приходит в голову мысль четко организовать код с помощью разделения на файлы. | Похожие файлы группируются в папку. | Каждый физический файл предназначен для чего-то одного, например, служит для объявления одного класса или для реализации одного функционала и т.д. | Организация кода на физическом уровне точно соответствует проекту, и, глядя на имена файлов и структуру папок, можно понять как спроектирована данная реализация. |  |
| Организация дерева исходников | Все в одной папке. | Простое разделение кода в логические подкаталоги. | Нет "круговых" зависимостей. Бинарники, либы, документация, билды, сторонний код -- все разложено в соответствующие папки. | Структура дерева исходного кода соответствует логической иерархии и организации кода в проекте. Глядя на имена файлов и структуру папок, можно понять как спроектирована данная система. | Разница между этим пунктом и предыдущим состоит в масштабе организации. Организация дерева исходников относится к всему комплексу продуктов, которые определяют систему. |
| читабельность кода | Односложные имена. | Хорошие имена файлов, переменных, классов, методов и т.д. | Нет длинных функций, а нестандартный код, багфиксы и допущения в коде поясняются комментариями. | Допущения в коде сопровождаются assert'ами, поток операций в коде естественный - нет глубокой вложенности условий или методов. |  |
| Безопасное программирование (defensive coding) | Не понимает данной концепции. | Проверяет все аргументы и ставит assert'ы на критические допущения в коде. | Убеждается, что проверил возвращаемое значение и что обрабатывает исключения в потенциально бажном коде. | Имеет свою собственную библиотеку помогающую в безопасном программировании, пишет юнит-тесты которые эмулируют сбои. |  |
| Обработка ошибок | Пишет код для "идеального" случая, когда все работает и нет сбоев. | Обработка ошибок в коде, который либо кидает исключение, либо генерирует ошибку. | Убеждается, что после того, как произошла ошибка/исключение, программа продолжает работать, а ненужные более ресурсы, коннекшоны и память были корректно освобождены обработчиком ошибки. | Пишет код так, чтобы определять возможные ошибки на раннем этапе, придерживается последовательной стратегии обработки исключений во всех слоях кода, разрабатывает общие принципы обработки исключений во всей системе. |  |
| Среда программирования (IDE) | В основном использует IDE для редактирования текста. | Способен эффективно пользоваться меню в IDE. Знает некоторые тонкости среды. | Для самых используемых функций среды знает горячие клавиши. | Написал свои макросы. |  |
| API | Часто нуждается в обращениях к документации. | Помнит самые часто используемые API. | Обширные и глубокие знания API. | Написал библиотеки, которые оборачивают API, для упрощения задач, которые наиболее часто встречаются. Эти библиотеки также часто восполняют пробелы в API. | Примером API может быть Java-библиотека, .Net фреймворк или API какого-либо приложения. |
| Фреймворки | Не использует никаких фреймверков за рамками основной платформы. | Знает, но не использует популярные фреймворки, доступные для его платформы. | Профессионально пользовался более чем одним фреймворком и хорошо разбирается в идиомах фреймворков. | Является автором фреймворка. |  |
| Требования | Понимает выставленные требования и пишет код в соответствии со спецификацией. | Задает вопросы касающиеся не рассмотренных в спецификации случаев. | Понимает картину в целом и предлагает дополнительные аспекты, которые должны быть описаны в спецификации. | Может предложить лучшие альтернативы и следовать выставленным требованиям, основываясь на собственном опыте. |  |
| Скрипты | Отсутствует знание скриптовых инструментов. | Batch-файлы/shell. | Perl/Python/Ruby/VBScript/Powershell. | Писал и публиковал повторно используемые скрипты. |  |
| Базы данных | Думает, что Excel - это база данных. | Знает основы баз данных, нормализацию, [ACID](http://ru.wikipedia.org/wiki/ACID" \t "_self), транзакции и может написать простые select'ы. | Может спроектировать хорошие нормализованные схемы БД, с учетом запросов, которые будут выполняться; умело использует представления, хранимые процедуры, триггеры и собственные типы данных. Понимает разницу между кластеризованными и не-кластеризованными индексами. Специалист в использовании [ORM](http://ru.wikipedia.org/wiki/ORM" \t "_self)-тулзов. | Может осуществлять администрирование БД, оптимизацию производительности БД, индексную оптимизацию, писать сложные select'ы, может заменить использование курсора вызовами функций sql, понимает как данные хранятся внутри, как хранятся индексы, имеет представление о том, как зеркалятся и реплицируются БД и т.д. Понимает как работает [двухфазный commit](http://en.wikipedia.org/wiki/Two-phase_commit_protocol" \t "_self). |  |
| Опыт | | | | | |
| Языки и профессиональный опыт | [Императивные](http://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) или [объектно-ориентированные](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) языки программирования | [Императивные](http://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), [объектно-ориентированные](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) и [декларативные](http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%BB%D0%B0%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) (SQL) языки программирования, дополнительный бонус - если понимает разницу между статической и динамической, слабой и строгой типизацией, [статически выводимыми типами](http://en.wikipedia.org/wiki/Type_inference" \t "_self). | [Функциональные языки программирования](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), дополнительный бонус - если знает, что такое "[ленивые вычисления](http://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BD%D0%B8%D0%B2%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F" \t "_self)", [каррирование, продолжения](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D1%80%D0%B8%D0%BD%D0%B3). | [Конкурентные](http://en.wikipedia.org/wiki/Concurrent_computing#Concurrent_programming_languages) (Erlang, Oz) и [логические](http://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5" \t "_self) (Prolog). |  |
| Годы профессионального опыта использования платформы | 1 | 2-3 | 4-5 | 6+ |  |
| Годы профессионального опыта |  | 2-5 | 6-9 | 10 |  |
| Знание предметной области | Не знает о понятии "[предметная область](http://en.wikipedia.org/wiki/Domain-specific_modeling" \t "_self)". | Работал хотя бы над одним продуктом в предметной области. | Работал над несколькими продуктами в одной и той же предметной области. | Эксперт своей предметной области, проектировал и реализовывал несколько продуктов/решений в ней, хорошо разбирается в ее сущностях и протоколах. |  |
| Знания | | | | | |
| Инструментарий | Ограничены используемой IDE ([VS.Net](http://ru.wikipedia.org/wiki/Visual_Studio" \t "_self), [Eclipse](http://ru.wikipedia.org/wiki/Eclipse) и т.д.) | Знает о некоторых альтернативах популярным стандартным инструментариям. | Хорошие знания редакторов код, отладчиков, различных IDE, open-source альтернативах и т.д. (Например, это может быть кто-то, кто знает большинство тулзов из [списка Скота Хансельмана](http://www.hanselman.com/blog/ScottHanselmans2007UltimateDeveloperAndPowerUsersToolListForWindows.aspx" \t "_self).)Использует [ORM средства](http://wiki.geeklog.net/wiki/index.php/Introduction_to_Object_Relational_Mapping_%28ORM%29_Tools). | Написал свои инструментарии и скрипты, дополнительный плюс - если эти скрипты были опубликованы. |  |
| используемые языки | Процедурные или объектно-ориентированные. | Процедурные, объектно-ориентированные и декларативные (SQL), дополнительный бонус за понимание статической vs динамической типизации, сильной vs слабой типизации и static inferred types. | Функциональные, дополнительный баллы за понимание lazy evaluation, карринга, continuations | Конкурентные (Erlang, Oz) и логические (Prolog) |  |
| Знание codebase | Никогда не смотрел [code base](http://ru.wikipedia.org/wiki/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B0" \t "_self). | Имеет общее представление о расположении кода и о том, как его собрать. | Хорошие рабочие знания code base, реализовывал несколько багфиксов и, может быть, некоторые маленькие фичи. | Реализовал несколько больших фич в code base и может легко описать изменения, требуемые для реализации большинства фич или багфисков. |  |
| Знание новейших технологий | Не слышал о новейших технологиях. | Слышал о новейших технологиях в своей области. | Слышал о новейших технологиях в своей области. Скачивал alpha/preview/CTP/beta-версии и читал некоторые стати и руководства на эти темы. | Пробовал сделать что-либо сам и используя preview-версию сбилдил свою программу. Дополнительный плюс - если сделал свое решение доступным для других. |  |
| Знание внутренних аспектов платформы | Нулевые знания внутренних аспектов платформы. | В основном, знает как работает платформа внутри. | Имеет глубокие познания внутренних аспектов платформы и может обрисовать, как платформа превращает программу в исполняемый код. | Написал свои тулзы для расширения возможностей платформы или для извлечения дополнительной информации о работе платформы. Например, дизассемблер, декомпилятор, отладчик и т.д. |  |
| Книги | Серии книг "... за 21 день", "... за 24 часа", "... для чайников". | [Совершенный код](http://oz.by/books/more.phtml?id=1015206&partner=devby)  [Не заставляйте меня думать!](http://oz.by/books/more.phtml?id=102668&partner=devby)  [Регулярные выражения](http://oz.by/books/more.phtml?id=102549&partner=devby) | [Приемы объектно-ориентированного проектирования. Паттерны проектирования](http://oz.by/books/more.phtml?id=101783&partner=devby)  [Человеческий фактор: успешные проекты и команды](http://oz.by/books/more.phtml?id=1015771&partner=devby)  [Жемчужины программирования](http://oz.by/books/more.phtml?id=103612&partner=devby)  [Algorithm Design Manual](http://www.ozon.ru/context/detail/id/1830000/)  [Программист-прагматик. Путь от подмастерья к мастеру.](http://oz.by/books/more.phtml?id=1031721&partner=devby)  [Мифический человеко-месяц или как создаются программные системы](http://oz.by/books/more.phtml?id=101035&partner=devby) | [Structure and Interpretation of Computer Programs](http://www.ozon.ru/context/detail/id/1831038/)  [Concepts Techniques and Models of Computer Programming](http://www.ozon.ru/context/detail/id/1829832/)  [Искусство программирования](http://oz.by/books/more.phtml?id=101588&partner=devby)  [Database systems , by C. J Date](http://www.ozon.ru/context/detail/id/1829471/)  [Thinking Forth](http://www.amazon.com/Thinking-Forth-Leo-Brodie/dp/0976458705/ref=pd_bbs_sr_1?ie=UTF8&s=books&qid=1216551535&sr=1-1)  [Little Schemer](http://www.amazon.com/Little-Schemer-Daniel-P-Friedman/dp/0262560992/ref=pd_bbs_sr_1?ie=UTF8&s=books&qid=1216551749&sr=1-1) |  |
| Блоги | Слышал о блогах, но не уделял им внимания. | Читает технические блоги, блоги о программировании и разработке ПО и регулярно слушает подкасты. | Ведет ссылочный блог, содержащий коллекции ссылок на полезные статьи и тулзы, которые [ссылки] он собирает. | Ведет блог, в котором содержатся его собственные понимание вопросов и соображения на тему программирования. |  |